MVA Microsoft Virtual Academy

MTA

Software Testing Fundamentals

Microsoft

Thomas Dawkins | Senior Product Manager
Erik Downing | Senior Software Development Engineer in Test (SDET)

Take the Exam

98-379 – Software Testing Fundamentals

- Certiport
  - http://www.certiport.com
- Prometric
  - http://www.prometric.com

Join the MVA Community!

Microsoft Virtual Academy

- Free online learning tailored for IT Pros and Developers
- Over 1M registered users
- Up-to-date, relevant training on variety of Microsoft product

http://www.microsoftvirtualacademy.com

Course Topics

Software Testing Fundamentals

01 | Testing Fundamentals
- 1.1 Software Testing
- 1.2 Software and Hardware Components
- 1.3 Fundamentals of Programming
- 1.4 Application Lifecycle Management

02 | Testing Methodologies
- 2.1 Testing Techniques
- 2.2 Testing Levels
- 2.3 Testing Types

03 | Creating Software Tests
- 3.1 User Centric Tests
- 3.2 Software Testability
- 3.3 Creating Test Plan Components
- 3.4 Feature Testing
- 3.5 Appropriately Scoped Test Cases

04 | Managing Software Testing Projects
- 4.1 Testing Milestones
- 4.2 The Agile Process
- 4.3 Working with Distributed Teams
- 4.4 Test Reports

05 | Working with Bugs
- 5.1 Detecting Software Defects
- 5.2 Logging Bugs
- 5.3 Managing Bugs

06 | Automating Software Test
- 6.1 Test Automation
- 6.2 Test Automation Strategies
- 6.3 Writing Automation Tests
- 6.4 Managing Test Scripts

MVA Microsoft Virtual Academy

06 | Automating Software Test

Microsoft

Thomas Dawkins | Senior Product Manager
Erik Downing | Senior Software Development Engineer in Test (SDET)

Module Overview

1. Understanding Test Automation
2. Test Automation Strategies
3. Writing Automation Tests
4. Managing Test Scripts

06 | Automating Software Test

6.1 Understanding Test Automation

Section Overview

In this section, we will review the following:

- Benefits
- Candidates for automation
- Automation process

Guiding Questions

What is the difference between an automated test and a manual test?

What are the benefits of automated testing?

What are the names of two types of tests that can be automated?

Automated Tests

An automated test can be thought of as a set of steps that a computer may run programmatically to test the functionality of the software.

Automated testing complements, but does not replace, manual testing.

Automated tests must be created by a developer, while manual testing can be conducted by someone without programming skills.

Benefits of Automated Testing

Automated tests help maintain stability and help find regressions that might occur because of code changes.

Automated tests can be run unattended.

Automated tests are software applications and can be designed and composed of other reusable code. This makes automated tests flexible and maintainable.

Automation can be run on multiple configurations by using Microsoft Test Manager.

Code coverage metrics can be gathered when automated tests are run.

Potential Drawbacks of Automated Testing

When code is changed or refactored, there may be cascading effects that will require a corresponding effort to change the affected automated tests.

There may be “psychological impact” on your team if code changes cause many tests to fail.

There may be a false sense of security when all tests pass if the test cases are not testing for the correct conditions.

Common Automated Tests

Some types of tests are almost always automated:

Unit Tests: In test-driven development, unit tests are created first; then developers write code to pass those unit tests.

Load Tests: Generating a heavy load would be difficult for manual testers, but automated tests can impose a great workload with little manpower.

Continuous Integration Tests: You can use continuous integration with Microsoft®Visual Studio® Application Lifecycle Management (ALM) to help ensure that whenever code is developed and checked

Other Potential Candidates for Automation

Configuration Testing: Testing on multiple installed environments can be a very laborious task. Microsoft Test Manager provides capabilities of running test suites on different configurations by using virtual machines or physical machines.

User Interface Tests: Visual Studio ALM has capabilities of creating automated tests directly for the user interface.

Installation Tests: You can use the lab capabilities of Microsoft Test Manager to set up a group of configurations that you can use to verify whether the

Barriers to Automation

The benefits of automation must be weighed against the costs. A few considerations when determining the amount of automation include:

- Creating automation requires a subset of the test team to learn how to write code—or requires hiring developers just to be on the test team.
- Code that is changing frequently is a moving target and will have cascading effects into test automation code because it will also need to be changed.
- Code that is tightly bound to the user interface is difficult to test because it may require the user to interact with the user interface controls.

Automation Process

You might begin your testing cycle by creating a manual test case that you decide later is a good test to automate. You want to be able to continue to run that test as part of a test plan.

1. Create an automated test and associate it with a test case using Visual Studio.
2. After you create your automated test, you must check in your test project that contains the automated test and make sure that the test project is part of your build definition.
3. Then you associate the automated test with a test case that has been added to a test plan that uses this build.

Section Questions

- ✓ What is the difference between an automated test and a manual test?
- ✓ What are the benefits of automated testing?
- ✓ What are the names of two types of tests that can be automated?

06 | Automating Software Test

6.2 Test Automation Strategies

Section Overview

In this section, we will review the following:

- Logging
- Automation priority

Guiding Questions

What feature enables automatic logging in Visual Studio?

Logging Data with IntelliTrace

IntelliTrace keeps a log of the key events as your tests and code execute in debug mode, and also logs variable values at those events.

You can step back through the history of execution before the test failed, inspecting the values that were logged. This is particularly helpful when working on bugs that are difficult to replicated.

To enable IntelliTrace, on the ***Visual Studio Debug*** menu, choose Options and Settings, IntelliTrace. You can also vary the settings to record more or less data.

Prioritizing Automation Efforts

Creating automated unit tests is the first priority for any test-driven development team—developers should be creating test cases before they build any methods or classes.

Likewise, integration tests become the top priority when units are combined into components.

When deciding which manual tests should be automated, consider the following:

- ✓ **How often will the test be run?** Give priority to frequently used tests.
- ✓ **How often with is the code likely to change?** Frequently changed code will require the tests to be maintained as well, possibly minimizing the benefits of automation.
- ✓ **How much time can be saved by running the test**

Section Questions

- ✓ What feature enables automatic logging in Visual Studio?

Great MSPress Books

- How We Test Software at Microsoft
  - http://www.microsoft.com/learning/en-us/boo 240&locale=en-us

Alan Page, Ken Johnston, RJ Rollison

06 | Automating Software Test

6.3 Writing Automation Tests

Section Overview

In this section, we will review the following:

- Logic
- Error Handling
- Commenting
- Virtual Machines

Guiding Questions

How is a virtual machine different from a physical machine?

What is the name of a condition test used to check logic in debug builds?

What happens to these condition tests in a release build?

Testing Logic with Assertions

An ***Assertion*** is a statement that tests a condition during program execution. If the assertion is true, no action occurs; if the assertion is false, the assertion fails.

In a debug build, a failed assertion causes the program to enter break mode, making it easier to examine the nature of the problem.

In a release build, assertions are ignored.

When you use an assertion, make sure that any code inside the assertion block does not change the results of the program if the assertion is removed. In other words, it should be only non-essential code.

- ✓ Otherwise, you might accidentally introduce a bug that only shows up in the release version of your program

Errors in Testing

Because automated tests consist of code, it is possible for the tests themselves to have errors.

If a test fails to run, you can investigate the failure by checking the test environment; this includes the way the test is set up and the settings in the active test settings.

In some cases, such as those related to deployment, failures are independent of test type. In other cases, the test type determines how and what to investigate.

Errors that involve tests are reported to you at either of two levels:

- ***Test-level errors.*** An example of a test-level error is a test timeout error, which occurs if the test's timeout limit is reached.
- ***Run-level errors.*** An example of a run-level error is a run timeout

Test Comments

While you run a test using ***Test Runner***, you have the option to add comments to the test steps.

The comments can be used to relay specific details that you want to communicate to other team members regarding unexpected behavior, or suggestions about what to look for while you test an application.

These comments can be useful as supporting material related to bugs, for isolating code defects, or for refining user interface designs.

Virtual Machines

In general, a computer system uses a single operating system. Since developers may want to test an application across many different operating systems for compatibility, this would necessitate having multiple computer system with various configurations, a potentially impractical approach.

However, you can set up a software environment that emulates an entire separate computer system by handling instructions in the same way a native processor (or “physical machine”) would handle instructions. This is called a virtual machine (VM).

By setting up a VM, you can test a different configuration without purchasing an additional computer.

Modern computer systems may even be capable of running multiple VM’s simultaneously.

Section Questions

- ✓ How is a virtual machine different from a physical machine?
- ✓ What is the name of a condition test used to check logic in debug builds?
- ✓ What happens to these condition tests in a release build?

06 | Automating Software Test

6.4 Managing Test Scripts

Section Overview

In this section, we will review the following:

- Smoke test
- Build verification test
- Lab Management

Guiding Questions

What are two types of lab environments?

What does BVT stand for?

What type of machines can a SCVMM environment manage?

Smoke Testing

In software, the term ***smoke testing*** describes the process of validating code changes before the changes are checked into the product’s source tree.

After code reviews, smoke testing is the most cost effective method for identifying and fixing defects in software.

Smoke tests are designed to confirm that changes in the code function as expected and do not destabilize an entire build.

In short, ***smoke tests*** focus on the features of the

Build Verification Tests (BVT)

A smoke test is also known as a Build Verification Test (BVT)

In this case, it gets its name because it is typically executed after a daily (or other “short term”) build.

The following steps are necessary to execute BVTs in Microsoft® Visual Studio®:

- ✓ Check in the newly created/modified test(s)
- ✓ Create a build definition
- ✓ Set up any necessary test agents
- ✓ Use Team Explorer to queue up and start the test(s)

Lab Environments

A thorough testing strategy includes tests conducted across multiple systems, perhaps with different hardware and operating systems.

Rather than manually re-configuring each machine for each set of tests, Visual Studio Lab Management handles configuration, deployment, and testing.

A ***lab environment*** is a collection of computers that are managed as a single unit, and on which you deploy the system under test along with test software.

Advantages of a Lab Environment

The results of tests can be shown on charts that relate them to system requirements.

Lab Manager automatically installs test agents on each machine, enabling test data to be collected.

You can view the consoles of the machines through a single viewer, switching easily from one machine to the other.

Lab environments manage the allocation of machines to tests for reasons that include preventing two team members from mistakenly assigning the same machine

Two Types of Lab Environments

Standard lab environment, which can contain a mix of virtual and physical machines.

SCVMM environment, which can only contain virtual machines that are controlled by System Center Virtual Machine Manager (SCVMM).

Features Available in SCVMM Environments

Environment snapshots: Environment snapshots contain the state of a lab environment, so you can quickly restore a clean environment, or save the state of an environment that has been modified.

Network isolation: Network isolation allows you to simultaneously run multiple identical copies of an SCVMM environment without computer name conflicts.

Virtual machine templates: A virtual machine template is a virtual machine that has had its name and other identifiers removed. This allows you deploy multiple copies of a virtual machine in the same environment, or multiple environments, and then run the virtual machines simultaneously.

Stored Virtual Machines: A virtual machine that is stored in

Supported Operating Systems for VM

- Windows® XP SP3 and later versions
- Windows Vista®
- Windows Server® 2003
- Windows Server® 2008
- Windows Server® 2008 R2
- Windows Server® 2012
- Windows 7
- Windows 8 or later version

Section Questions

- ✓ What are two types of lab environments?
- ✓ What does BVT stand for?
- ✓ What type of machines can a SCVMM environment manage?

Additional Resources for this Module

MSDN Software Testing Resources	
Walkthrough: Creating and Running a Load Test Containing Web Performance Tests	http://msdn.microsoft.com/en- us/library/dd537628.aspx
Walkthrough: Creating a Simple Web App	http://msdn.microsoft.com/en-us/library/ms243156.aspx
Walkthrough: Recording and Running a Web Performance Test	http://msdn.microsoft.com/en-us/library/ms182551.aspx
Using Code Coverage to Determine How Much Code is being Tested	http://msdn.microsoft.com/en- us/library/dd537628.aspx
Unit Testing: Testing the Inside	http://msdn.microsoft.com/library/jj15934 0.aspx
Debug Your App by Recording Code Execution with IntelliTrace	http://msdn.microsoft.com/en-us/library/vstudio/dd264915.aspx
Walkthrough: Creating and Running a Generic Test	http://msdn.microsoft.com/en-us/library/ms182626.aspx
Assertions in Managed Code	http://msdn.microsoft.com/en-us/library/ttcc4x86(v=vs.110).aspx
Lab Environments	http://msdn.microsoft.com/en- us/library/jj159341.aspx
Troubleshooting Test Execution	http://msdn.microsoft.com/en- us/library/ms182478.aspx
Walkthrough: Managing Tests Using Lists and Properties	(http://msdn.microsoft.com/en-US/library/ ms182466(v=vs.100).aspx
Types of Performance Testing	http://msdn.microsoft.com/en- us/library/bb924357.aspx

Know your stuff? Get Certified!

- Microsoft Certified Solutions Developer (MCSD) for Application Lifecycle Management
  - http://www.microsoft.com/learning/en-us/mcsd-application-lifecycle-management.aspx

- Microsoft Learning Partners—Learn from the Pros!
  - http://aka.ms/CPLS
  - Find a Class: http://aka.ms/ClassLocator

©2013 Microsoft Corporation. All rights reserved. Microsoft, Windows, Office, Azure, System Center, Dynamics and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.